HTML5 may provide vital link for friendly future mobile applications

By LTC Gregory Motes

An important thread that has existed in the background of the U.S. Army Training and Doctrine Command's Connecting Soldiers to Digital Applications program has been the desire to create applications that fell into the so-called "device agnostic" category.

As TRADOC leaders continue evaluating the role and viability of mobile devices in learning, training, and operational environments, a few goals have emerged. For example, we want to foster an atmosphere that sets priorities that limit duplication when creating applications that have financial and practical implications. Simply put, the best scenario is to be able to build training content once and to have it work across a maximum number of devices in both on-line and off-line states. To many, this would be accomplished with the maturation of HTML5.

The best scenario is to be able to build training content once and to have it work across a maximum number of devices in both on-line and off-line states. To many, this would be accomplished with the maturation of HTML5.

To fully grasp the opportunity, one needs to be aware of the history of HyperText Markup Language. In the early 1990s, Tim Berners-Lee created a new protocol called HyperText Transfer Protocol and a new text format markup language based on the Standard Generalized Mark-up Language. HTML notably added hyperlinks with an anchor element that carried an HREF attribute.

Over the next decade, HTML standardization, open standards and adoption engendered the modern Web site. Subsequent scripting languages, such as JavaScript, created a powerful tool for Web developers to create robust information and interactive Web sites. The power of HTML was its ability to interoperate on multiple browsers and platforms, providing users a similar experience without regard to their environment.

Yet, even this was not agnostic, as can be attested by people who chose to adopt new browsers and versions. Backward and forward compatibility created difficult challenges for developers - often exasperated in the Enterprise setting where adoption of new technologies had to undergo interoperability and security testing. The practical result within the military has been that systems and applications are often at least one version behind the current consumer offerings.

The crescendo of change that arguably started in 2007 when Apple announced the first generation iPhone has been growing louder as leaders and Soldiers have experienced the power of Smartphone's in their personal lives. The gap between potential capabilities offered by Android and iPhone/iPad and the technology presented in Army classrooms continues to increase as TRADOC examines fiscally sound ways to use the new medium. This has put training developers into a familiar "chicken versus egg" contest that seems to accompany many advances in technology. The question is raised about how to get funding for technologies that haven't previously been funded and how much time and existing resources should be used for pilot programs.

This evolution of technology has been particularly tricky due to the programming barriers that exist with creating native applications in iOS and Android. During the early stages of the CSDA program, a perception existed that developing Smartphone apps was not that difficult. This was likely based on the fact that there had been over 100,000 apps developed and published for the iTunes apps store after just one year. "How hard can it be?" Similarly, the new Android market (at the time) was also receiving a steady influx of apps. Still, the leap from programming in easier languages like HTML and Flash's ActionScript to object oriented languages like Objective C or Java was still a considerable one.

It should be noted that the rise of the native

(Continued on page 32)

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2012		**00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
HTML5 may provide vital link for friendly future mobile applications	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Army Signal Center of Excellence and Fort Gordon,Army Communicator,Signal Towers (Building 29808), Room 713,Fort Gordon,GA,30905-5301**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**3**	

(Continued from page 31)

app was unpredictable, even to Apple. Early documentation of the iPhone championed the rise of the web app and in fact did not have a substantial plan for native app development through a software development kit. The Internet still teems with web sites that gush about the potential for web apps using the new (2007) iPhone. Many of those sites, interestingly enough, return page not found results because, as it turns out, the web app was not the dawn of a new future for means to connect users to information and tools. Instead, in the shadows rose a community of hackers who began to reverse engineer certain parts of the iPhone operating system application programming instructions and created an underground market to distribute apps that perform functions not provided by the iPhone. In fact, Apple did not even have an app store, until 1 year after the release of the first version of the iPhone, and still hyped web apps.

At some point, Apple realized that there was a market for third party apps on iPhone - which is really no different from their allowing a mechanism to put third party apps on Mac - and started to court a developer community that quickly blossomed. The result was that production of web apps stalled, giving way to native apps. Yet, even the success of the native app, coupled with success of native Android apps, there is still an underlying clamoring for a standard language that can be written once and deployed everywhere. Native app development can range from cheap to expensive and from agile to cumbersome. Since the military has yet to choose Android or iPhone as its sole target for all apps (and likely doesn't intend on choosing one or another as the only solution for all cases), organizations that are interested in pursuing content delivery through a Smartphone are stuck with either choosing or having to develop for multiple platforms.

TRADOC can envision many different use cases for applications. One is the student in the classroom, the other is the student in the field, another is a student who is preparing to come to a professional military education course, and another is a graduate of a course. In an operational sense, Soldiers could use Smartphones to access information in environments that range from the orderly room, to the motor pool, to the clinic, and then into the field, whether it is training or in a combat environment. This makes isolating the environment an important part of the narrative. So, for a moment, just consider the Soldier who is preparing to attend formal military instruction and is required to take some training prior to arrival. This Soldier/student likely has access to a PC, but could also have access to a Smartphone or a tablet.

Prior to discussions leading to the Army Learning Model 2015, TRADOC's training developers were likely targeting just the PC and trying to account for multiple browsers if they were interested in reaching more users -- though most likely just targeting Internet Explorer. Even with creating modules in Flash that target a specific version of IE, it is a problem. At press time, the author of this article is working on a government computer that has Internet Explorer version 7.0.6 and Adobe Flash Player 11.2.202.228 Users at home likely have Internet Explorer version 9 (or are using Chrome, Firefox or Safari) and Flash Player 11.2.202.233. These discrepancies can return unpredictable results, creating the potential for interoperability, yet in a risk averse, low cost environment, these limitations are tolerated.

Modern Smartphone operating systems complicate matters even more. The goal of Smartphone "agnostic" applications has been an illusion, yet is still an idealistic goal that holds interest in communities like TRADOC that are trying to write once, deploy everywhere. It is with that, where HTML5 holds the promise of deploying content that will work across a maximum number of devices with limited interoperability issues, and in many ways filling the role that Flash has played in the past in the browser. As Flash has fallen out of favor as the de facto standard for interactive content on mobile devices - due largely to stability and power issues - HTML5 was increasingly presented as the alternate.

Between 2004 and 2009 groups within the World Wide Web Consortium developed positions and requirements for future hypertext application technologies, ultimately leading to the progression of a standard for HTML5. Some of the key components included improved graphics support with canvas and scalable vector graphics, wider multimedia support without the use of plug-ins, geo-location support within a web applications, and an application cache that could provide offline storage for apps. As an example, prior to HTML5, users could not draw on the web without the use of tools like Flash and Silverlight, but the ability to embed SVG into the document object model increase the capabilities presented to users natively in their browsers. HTML5 also offers a number of new APIs that will extend features that had to previously be programmed in other languages, including drag and drop, flexible parsing, system and directory access, and more robust error handling.

Yet for all of the interest, developers and users will still have to wait until 2014 for the entire HTML5 specification to be declared. It can be argued that the deliberate pace of implementation is prudent to come up with a lan-

guage that may bear the standard for 15 to 20 years, as its predecessor will have done, but is a source of frustration for those that are looking at it to be a viable alternative to native application development. Some browsers and applications already recognize HTML5 components. Notably, YouTube has a HTML5 implementation of its video player that tests fairly well despite certain restrictions. It is expected that browsers will gently include many of the HTML5 standards prior to the full specification, as is already available in the Webkit browsers that support certain HTML5 media tags already.

This interstitial period between desired effects and full scale implementation will continue to be bolstered by native app development, with an expected steady increase in applications that rely on HTML5. For mobile applications, several programs and development environments have gained momentum in allowing developers the opportunity to code in one integrated development environment and then have separate code compiled for different mobile operating systems. Among these, PhoneGap and Titanium are two notable efforts that have attempted to infiltrate the niche of developers that are trying to decrease the amount of work it takes to get an application onto multiple platforms. While both of these have certain strengths, the developer essentially has to learn another programming language (the appropriate API) and will likely experience a letdown trying to get the native look and feel they desire.

In the meantime, we have suggested that device "agnostic" apps are unlikely to appeal to the users who are infatuated with the user interface elements of their devices. Users on a PC will expect to have access to certain features by using the mouse right click or hovering over UI elements. Even though there have been some attempts at using a "long press" or a "two finger press" to bring up comparable menu options, the concept of hovering does not translate to touch screens. Furthermore, Android users have different expectations and anticipations than iPhone users. The most notable has been with the hardware menu button, where Android users will expect to be able to press that button during the operation of an app and be presented with additional menu options. This button does not exist on an iPhone, instead it is replaced by software buttons and tab-based navigation that is instantly familiar to its users. Blackberry's Playbook encourages developers to use gestures generated from the bezel to access additional information, which is again a concept that is unavailable for other devices. So, as developers are working toward creating an application that will work on a desktop pc, Android Smartphone, iPhone, Android tablet, iPad, or devices such as the Playbook, different user interfaces and programming logic will still exists.

HTML5 may eventually allow for an effective solution for developers to reach common denominators on devices and even account for different hardware capabilities using JQuery and CSS. Still, this promise is not going to eliminate the work of presenting standardized content on fragmented hardware while maximizing the potential of the leading tiers of consumer devices.

LTC Gregory Motes ***is an Armor Officer in Functional Area 53 Information Systems Management and is a doctoral candidate in Instructional Technology, examining situation awareness through a combat alert notification system. Over his career he has held a variety of leadership and staff assignments including tank company commander, MNF-I theater information assurance policy officer, division automation management officer and division chief at the School of Information Technology. He deployed his tank company to KFOR 2B, served in CFLCC and MNF-I for 16 months, and has 3 overseas assignments. He recently led the U.S. Army Mobile Applications Branch at Fort Gordon, Ga., which had a team of officers and contractors develop nearly 100 applications that received numerous awards and over 1,500,000 downloads on iTunes and Google Play.***

Acronym QuickScan

API – Application Programming Interface
CSDA – Connecting Soldiers to Digital Applications
DOM – Document Object Model
HREF – HyperText Reference
HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol
IDE – Integrated Development Environment
IE – Internet Explorer
iOS – iPhone Operating System
PC – Personal Computer
SGML – Standard Generalized Markup Language
SVG – Scalable Vector Graphics
TRADOC – U.S. Army Training and Doctrine Command
UI – User Interface
W3C – World Wide Web Consortium